<https://leetcode.com/problems/shortest-distance-to-a-character/>

Given a string s and a character c that occurs in s, return *an array of integers* answer *where* answer.length == s.length *and* answer[i] *is the* ***distance*** *from index* i *to the* ***closest*** *occurrence of character* c *in* s.

The **distance** between two indices i and j is abs(i - j), where abs is the absolute value function.

**Example 1:**

Input: s = "loveleetcode", c = "e"

Output: [3,2,1,0,1,0,0,1,2,2,1,0]

Explanation: The character 'e' appears at indices 3, 5, 6, and 11 (0-indexed).

The closest occurrence of 'e' for index 0 is at index 3, so the distance is abs(0 - 3) = 3.

The closest occurrence of 'e' for index 1 is at index 3, so the distance is abs(1 - 3) = 2.

For index 4, there is a tie between the 'e' at index 3 and the 'e' at index 5, but the distance is still the same: abs(4 - 3) == abs(4 - 5) = 1.

The closest occurrence of 'e' for index 8 is at index 6, so the distance is abs(8 - 6) = 2.

**Example 2:**

Input: s = "aaab", c = "b"

Output: [3,2,1,0]

**Constraints:**

* 1 <= s.length <= 104
* s[i] and c are lowercase English letters.
* It is guaranteed that c occurs at least once in s.

**Attempt 1: 2023-04-02**

**Solution 1: Two Pass (30 min)**

class Solution {

public int[] shortestToChar(String s, char c) {

int len = s.length();

int[] result = new int[len];

// No need prev = Integer.MIN\_VALUE / 2, because

// the definition is 1 <= s.length <= 10^4

int prev = -10001;

for(int i = 0; i < len; i++) {

if(s.charAt(i) == c) {

prev = i;

}

result[i] = i - prev;

}

// No need prev = Integer.MAX\_VALUE / 2, because

// the definition is 1 <= s.length <= 10^4

prev = 10001;

for(int i = len - 1; i >= 0; i--) {

if(s.charAt(i) == c) {

prev = i;

}

result[i] = Math.min(result[i], prev - i);

}

return result;

}

}

Time Complexity: O(N), where N is the length of S. We scan through the string twice.

Space Complexity: O(N), the size of ans.

**Refer to**

<https://leetcode.com/problems/shortest-distance-to-a-character/editorial/>

#### **Approach #1: Min Array [Accepted]**

**Intuition**

For each index S[i], let's try to find the distance to the next character C going left, and going right. The answer is the minimum of these two values.

**Algorithm**

When going left to right, we'll remember the index prev of the last character C we've seen. Then the answer is i - prev.

When going right to left, we'll remember the index prev of the last character C we've seen. Then the answer is prev - i.

We take the minimum of these two answers to create our final answer.

class Solution {

public int[] shortestToChar(String S, char C) {

int N = S.length();

int[] ans = new int[N];

int prev = Integer.MIN\_VALUE / 2;

for (int i = 0; i < N; ++i) {

if (S.charAt(i) == C) prev = i;

ans[i] = i - prev;

}

prev = Integer.MAX\_VALUE / 2;

for (int i = N-1; i >= 0; --i) {

if (S.charAt(i) == C) prev = i;

ans[i] = Math.min(ans[i], prev - i);

}

return ans;

}

}

**Complexity Analysis**

* Time Complexity: O(N), where N is the length of S. We scan through the string twice.
* Space Complexity: O(N), the size of ans